charles river analytics

Charles River Analytics Inc.
625 Mount Auburn St.
Cambridge, MA 02138

www.cra.com

p: 617.491.3474
f: 617.868.0780

15 MAR 2010

Reference: Government Contract No. N00014-09-C-0050, "Enhancing Simulation-based Training Adversary Tactics via Evolution (ESTATE)"
Charles River Analytics Contract No. C08098

Subject: Contractor's Status Report: Quarterly Status Report #5
Reporting Dates: 12/15/2009 – 3/15/2010

Dear Dr. Hawkins,

The following is the Contractor's Quarterly Status Report for the subject contract for the indicated period. During this reporting period work has concentrated on Task 2: Develop Mitigation Methods, Task 3: Enhance Adaptation Techniques, and Task 4: Develop Trainee Model Processing.

# 1. Summary of Progress

During this reporting period, we researched, implemented, and evaluated a method for coevolution of strategy based game players and game challenges to prepare for use in ESTATE-based training. We also mined the existing MoneyBee data set with our academic partner, Brandeis University.

## 1.1 Task 2: Develop Mitigation Models and Task 3: Enhance Adaptation Techniques, Coevolution implementation and testing.

The purpose of Task 2 is to mitigate the effects of coevolutionary pathologies on trainee progress with ESTATE. Applying coevolution techniques often fails to generate the desired goal of a continuous learning process leading to ever-improving individuals. Recent research has begun to identify and define the pathologies hindering success. With the assistance of our university partner, we have begun to understand these pathologies, such as:

- Disengagement: Loss of competitive gradient causes improvement to stop occurring
- Cycling / Intransitivity: Oscillation back and forth between strategies instead of making progress
- Overspecialization / Focusing: Concentration in one area at expense of other areas
- Evolutionary Forgetting: Loss of useful trait from one generation to the next

# Report Documentation Page

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE | 2. REPORT TYPE | 3. DATES COVERED |
|---|---|---|
| **15 MAR 2010** | | **00-00-2010 to 00-00-2010** |

| 4. TITLE AND SUBTITLE | 5a. CONTRACT NUMBER |
|---|---|
| **Enhancing Simulation-Based Training Adversary Tactics Via Evolution (Estate)** | 5b. GRANT NUMBER |
| | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | 5d. PROJECT NUMBER |
| | 5e. TASK NUMBER |
| | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Charles River Analytics, Inc.,625 Mount Auburn St,Cambridge,MA,02138** | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSOR/MONITOR'S ACRONYM(S) |
| | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT
**Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT **unclassified** | b. ABSTRACT **unclassified** | c. THIS PAGE **unclassified** | **Same as Report (SAR)** | **20** | |

- Red Queen Effect: Changes which improve quality of a solution do not increase its selection probability due to changes to other coevolving solutions

By understanding these pathologies, we can begin to develop an adaptation system for tailored training challenges and strategies that features methods to monitor and mitigate them. Based on the many common pitfalls facing coevolving systems, we strongly believe that any approach that fails to address these competitive pathologies will be subject to failure.

During this reporting period we examined the use of the following techniques for mitigating coevolutionary pathologies.

- Capturing Informativeness – Mitigating "loss of gradient / disengagement" by identifying how solutions can inform (e.g., test) other competing solutions and maintaining them in the population based on this criteria
- Separation of Teacher and Learner Populations – Capturing informativeness by explicitly separating the population of strategies into two populations, one that informs the evolutionary algorithm on how the other one is doing.
- Memory Mechanisms – Improving upon standard elitism using "Hall of Fame" techniques to provide an growing external benchmark to compare newer potential solutions against older potential solutions, mitigating the "cycling" pathology

To capture informativeness and separate teacher and learner populations, we employ student-test coevolution. Our coevolution simulations consist of two populations. One population, the tests, represent the challenges that may be presented to the trainees. The other population, the students, represent strategies that the trainees may use to attempt the challenges. Because the purpose of student-test coevolution is to produce better students and the quality of the tests are only important in their ability to improve student performance, different selection strategies are used for students and tests. Students are chosen according to their ability to pass all of the current tests. The highest performing group of students are usually chosen for the next generation. Tests, however, are chosen according to their ability to inform on the students progress. A test that defeats all students is not as useful to the algorithm as one that defeats only half of the students. The second test provides information about which students are better, and thus should be generally preferred to the first test.

### *1.1.1 Selecting a Solution Concept for Student-Test coevolution*

Ficici (2004) identifies *solution concepts* as a method to analyze the relationship between the selection of individuals in coevolution and the meeting of the overall goals of the coevolutionary process. It indicates which individuals to keep for future populations; thus, a solution concept is a type of memory mechanism. A well functioning solution concept will drive the population towards the goals (e.g. being a better game player), while a poorly functioning solution concept will cause the population to flounder due to one or more coevolutionary pathologies.

A *monotonic* solution concept (Ficici, 2004) is one that causes the best individual in the population to drift no further from the goal, with some chance of evolving towards the goal. Thus the population monotonically increases its fitness according to the goal. A monotonic solution concept prevents the pathologies of cycling and intransitivity, evolutionary forgetting, and the

red queen effect from occurring during coevolution. To avoid these pathologies while still allowing execution within a reasonable time, we chose and implemented a method to approximate a monotonic solution concept for student-test coevolution.

Several general purpose monotonic solution concepts have been identified in the literature. Rosin (1997) identifies a solution concept that selects students that simultaneously maximize outcomes over all tests. However, we do not expect our challenges to allow a single, correct solution at each level. Ficici (2004) identifies a solution concept according to the Nash equilibrium, where no individual can individually change his strategy without decreasing his payoff, i.e. there is no individual incentive to change. The IPCA algorithm (De Jong, 2004) identifies a solution concept based on the Pareto-optimal equivalence set, a set that provides maximal trade-off between objectives. MaxSolve (De Jong, 2005) identifies a solution concept based on the maximum expected outcome of a student over the current population of tests. Finally, DECA (De Jong & Bucci, 2006) identifies a solution concept based on estimating the true problem dimensionality, the number of different objectives which must be maximized simultaneously.

Our criteria for selecting a solution concept was that 1) the solution concept performed well in practice and 2) the solution concept did not further constrain on the problem. ESTATE's effectiveness as an adaptive training environment will be increased the better a solution concept is able to perform. New challenges will be created more quickly, and they will be closer to the optimal Zone of Proximal Development of the trainee. Secondly, we did not wish to overly constrain the problem that our coevolutionary technique can address. Such constraints may prevent training of critical skills, and we wish for ESTATE to be applicable to as many skill sets as possible.

Performance comparisons between these algorithms (De Jong, 2005; De Jong et al., 2006), communications with authors (Bucci, 2010), and consultation with our academic partner, an expert in this area, led us to choose the MaxSolve solution concept as the best candidate for implementation and testing. MaxSolve has exhibited high performance on a number of different challenges, and it does not place any additional constraints on the problem.

### *1.1.2 Implementing MaxSolve and Challenge games*

To evaluate the performance of the MaxSolve solution concept for use in ESTATE, we implemented student-test coevolution, the solution concept, and several test problems. We leveraged our in-house evolutionary algorithms toolkit, EAToolkit, and expanded the toolkit to support competitive coevolution, in which individuals are scored according to one-on-one competitions, and student-test coevolution, in which a separate student and test populations are maintained. Tests are challenges that may be presented to a trainee, and students are strategies for overcoming challenges. Ideally, coevolution will result in challenges of increasing difficulty and strategies of increasing effectiveness.

We implemented the MaxSolve solution concept as described in (De Jong, 2005). To ensure a correct implementation and provide an assessment of performance, we implemented three separate games for coevolution testing.

The first game implemented was the discretized COMPARE-ON-ONE numbers game from the MaxSolve paper (De Jong, 2005). The numbers game is a simple game where the individuals attempt to increase the values of their vectors of real numbers. Both students and tests are

individuals with a vector of numbers. When a student attempts a test, the individual with the higher value in the slot with the test's highest value wins. This game is advantageous in that it provides a difficult test case for coevolution (because the simple mechanics are a black box to the coevolutionary algorithm) while remaining open to rapid analysis.

The second game implemented was the challenge tree game, intended to mimic the structure of a actual strategy space that may be input to ESTATE. A challenge tree is a complete *k*-ary tree of depth *d.* Each non-leaf node in the tree has *k* children, and the path from the root node to a leaf node is of length *d*. A number, *g*, of the leaf nodes are identified as goal states. Figure 1 is an example challenge tree with *k*=3, *d*=4, and *g*=4. A challenge tree can be played by beginning at the root and choosing a child node to move to until a leaf node is reached. If the leaf node is a goal state, then the game was won, else the game was lost. In student-test coevolution, tests are sub-trees (beginning nodes) of a larger challenge tree, and students are strategies consisting of <node, child-node> pairs specifying which child node to choose at each node.

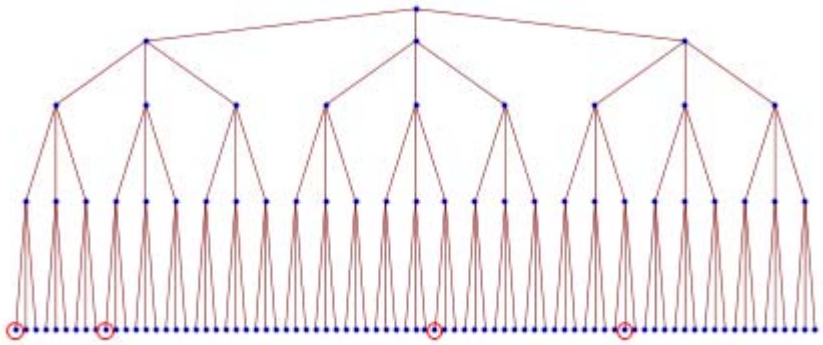


**Figure 1: An example challenge tree game with k=3, depth=4, and number of goals=4**

The third game implemented was the game of Nim. This game was intended to test the coevolution on an actual game that humans find challenging to play despite the existence of a relatively straightforward perfect strategy (Bouton, 1901). Nim is played with *n* heaps of stones of varying sizes. Each player takes turns selecting a heap, then picking up one or more stones from that heap. The player to pick up the last stone wins. Figure 2 is an example game of Nim with heap sizes 1, 2, 3, 4, and 5. In student-test coevolution, tests are initial game states, and students are strategies consisting of <state, action> pairs specifying which heap to select and how many stones to remove from this heap. Because Nim is a 2-player game, we pit the students against an automated player with the perfect strategy (most games can still be won, as the student makes the first move). This formulation is particularly restrictive because any deviation from the perfect strategy within the sub-game will result in a loss.

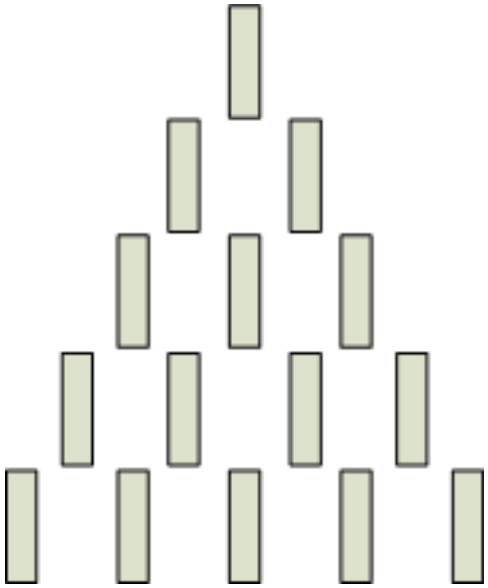

**Figure 2: Example game of Nim: <1,2,3,4,5>. Each row represents a heap. Players choose a row and remove one or more bars. The player to remove the last bar wins.**

### *1.1.3 Testing performance of MaxSolve coevolution*

The first test of our MaxSolve implementation was to reproduce the results of the original paper using the discretized COMPARE-ON-ONE game (De Jong, 2005). Figure 3 shows the results of this test. These results match those of the paper; MaxSolve is able to sustain continuous student improvement on three dimensions by maintaining a diverse set of tests. This confirms the results of the paper and supports our claim of correct implementation.

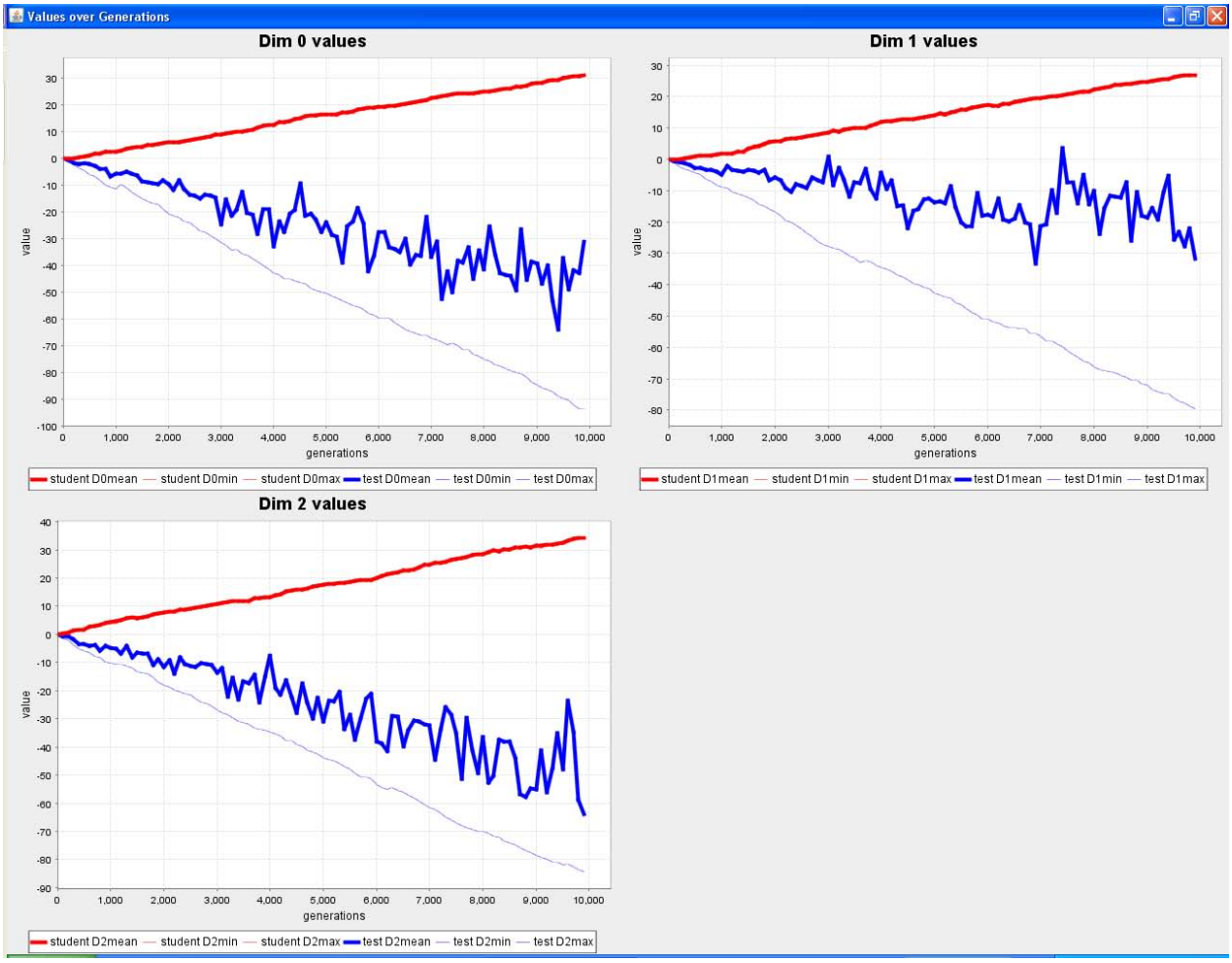


**Figure 3: MaxSolve implementation on the 3 dimensional discretized COMPARE-ON-ONE game. Student values (in red) increase steadily, and Test values (in blue) maintain a diverse set.**

The second test was to use MaxSolve in the challenge tree game. Figure 4 shows the results of challenge tree coevolution with k=3, d=8, g=10, and the mutation rate for students set to 1 allele (one node's strategy is changed for each child). The best students are able to find a goal state for 57% of the winnable nodes by generation 1000.

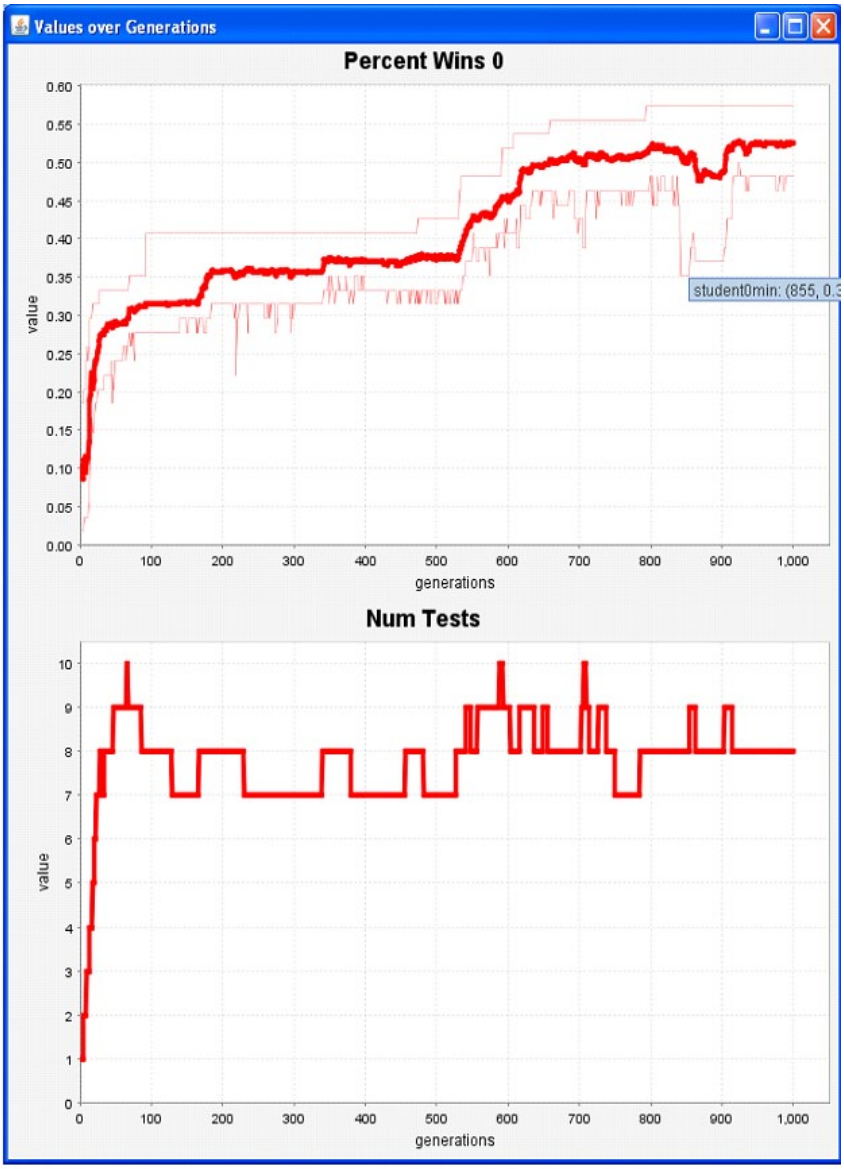


**Figure 4: Results of MaxSolve on the Challenge Tree game, k=3, d=8, g=10, mutation rate = 1 allele. The top graph shows the mean, minimum, and maximum percentage of winnable nodes that the student population is able to win, graphed by the population generation. As the coevolution progresses, the population improves its ability to win the game. The bottom graph shows the number of tests kept by MaxSolve.**

One of the issues in applying coevolutionary solutions to problems such as this is the tuning of algorithm parameters to improve performance. Parameters in this formulation of student-test coevolution are student mutation rate, test mutation rate, student crossover percentage (the percent of new individuals are created through crossover), test crossover percentage, student archive size, student population size, test population size, and initial population sizes. As an example of how these parameters may contribute to the effectiveness of the coevolution, Figure 5 shows the results of increasing the student mutation rate to 10 alleles. Here, the best students are able to find a goal state for 99% of the winnable nodes by generation 1000, a substantial

improvement over the previous run. To provide some insight into optimal parameter settings, we performed a sensitivity analysis MaxSolve coevolution of the COMPARE-ON-ONE game; the results are summarized in Section 1.1.4. These results show that MaxSolve can be effective in strategy domains such as those ESTATE may encounter, given proper parameter settings.

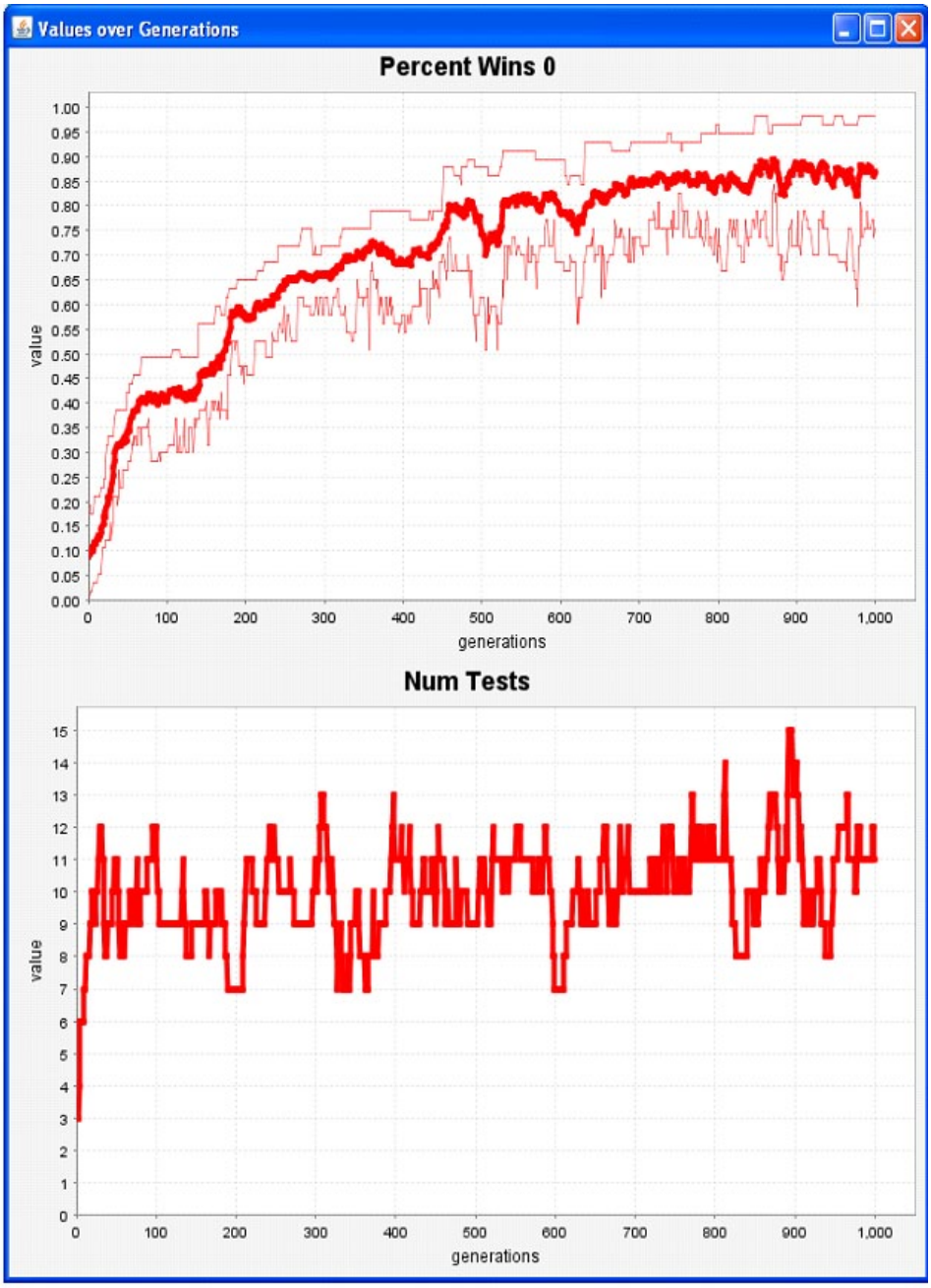


**Figure 5: Results of MaxSolve on the Challenge Tree game, k=3, d=8, g=10 (same as Figure 4), *mutation rate = 10 alleles*. The top graph shows the mean, minimum, and maximum percentage of winnable nodes that the student population is able to win, graphed by the population generation. As the coevolution progresses, the population improves its ability to win the game. The bottom graph shows the number of tests kept by MaxSolve.**

Neither the challenge tree game nor the COMPARE-ON-ONE game are difficult for humans to learn or solve, the next test was to apply MaxSolve to the Nim game to test performance on a larger problem that humans do not easily solve. In this regard, Nim is representative of the aims of ESTATE, to aid trainees in developing real world skills on difficult problems and tasks. Figure 6 shows the results of running MaxSolve coevolution on the Nim game with heap sizes = <3,3,3,3>. Here, coevolution is again successful in finding a winning strategy after about 600 generations; the best student is able to win against the perfect player at any winnable sub-game – it has evolved the perfect strategy.

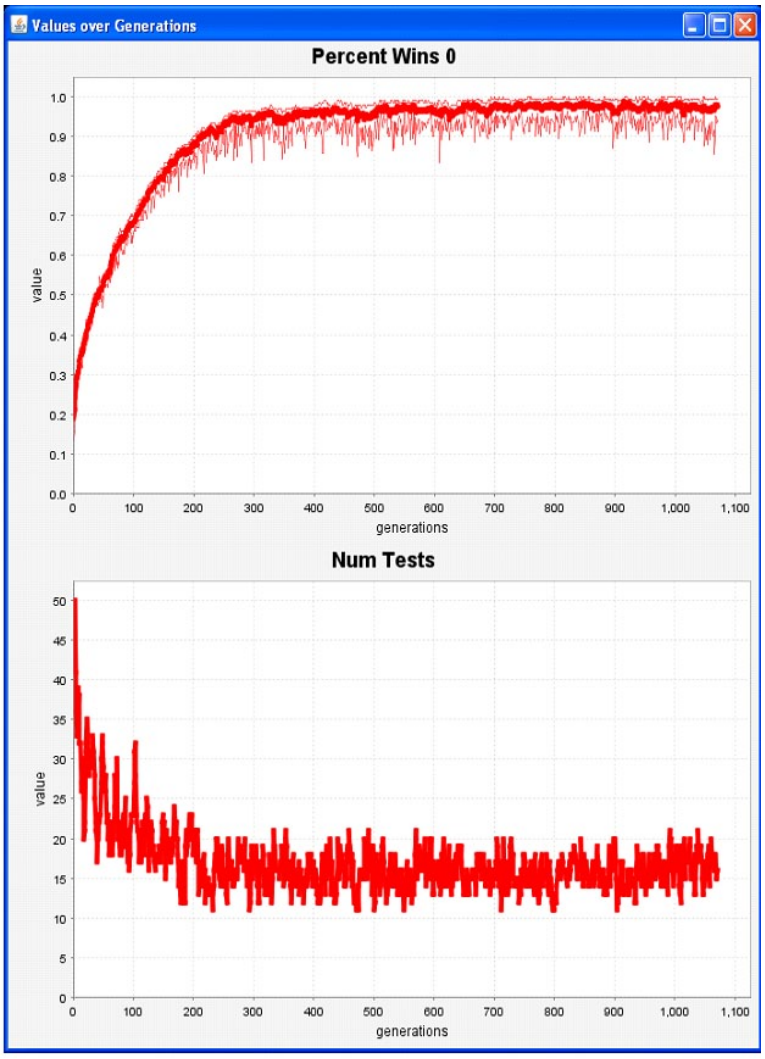


**Figure 6: Results of MaxSolve on the Nim game. Heaps = <3,3,3,3>. The top graph shows the mean, minimum, and maximum percentage of winnable nodes that the student population is able to win, graphed by the population generation. As the coevolution progresses, the population improves its ability to win the game. The bottom graph shows the number of tests kept by MaxSolve.**

Nim is made more difficult for coevolution by increasing the size of the piles. Figure 7 shows the results of running MaxSolve coevolution on Nim with heap sizes = <3,4,5,4>, more than doubling the size of the state space. Here, the coevolution takes much longer to converge on a

successful strategy, but the best player is still able to win 95% of the winnable games against the perfect player after 10,000 generations. These results show good performance of MaxSolve coevolution on a game that novice humans have difficulty winning consistently. This is a strong indication that our MaxSolve student-test coevolution will be able to make progress in domains that require non-trivial strategic formulations, such as those training domains that ESTATE targets.

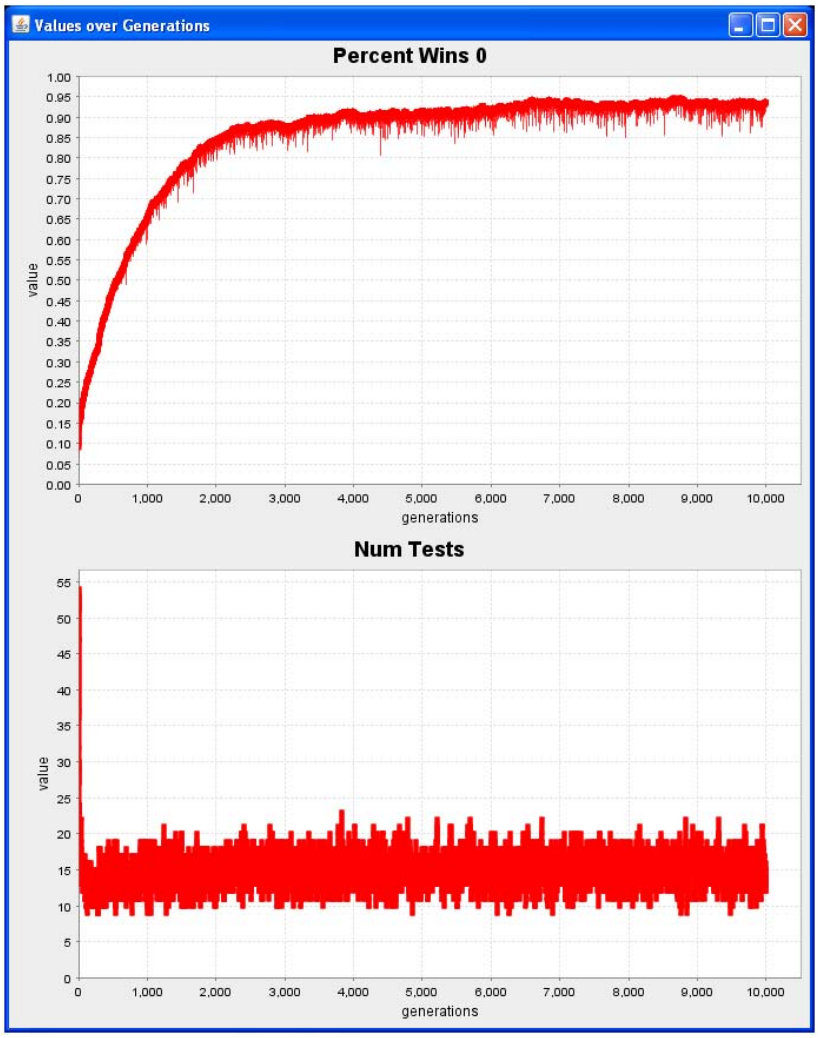


**Figure 7: Results of MaxSolve on the Nim game. Heaps = <3,4,5,4>. The top graph shows the mean, minimum, and maximum percentage of winnable nodes that the student population is able to win, graphed by the population generation. As the coevolution progresses, the population improves its ability to win the game. The bottom graph shows the number of tests kept by MaxSolve.**

Together, these results show strong support that MaxSolve can produce successful coevolution on a range of different, yet representative, problems.

### *1.1.4 Sensitivity Analysis of MaxSolve Coevolution*

One of the issues noted above in using coevolution is the tuning of algorithm parameters to improve performance. As the challenge tree example in Section 1.1.3 exhibits, choosing optimal parameters can make the difference between success and failure. Parameters in our MaxSolve student-test coevolution are student mutation rate, test mutation rate, student crossover percentage, test crossover percentage, student archive size, student population size, test population size, and initial population sizes. Also to be considered is the difficulty of the problem under consideration. Here, we perform a sensitivity analysis of problem dimensionality (number of simultaneous objectives, roughly a measure of difficulty), MaxSolve archive size, student mutation rate, test mutation rate, and crossover percentage for the discretized COMPARE-ON-ONE numbers game as defined in Section 1.1.2. The sensitivity analysis indicates how these parameters interact to produce a change in the result.

1700 samples of the parameter space were created using a Latin Hypercube design. Student archive size ranged from 10 to 160, dimensions ranged from 2 to 10, student and test mutation rates ranged from 0.05 to 0.75, student crossover percentage ranged from 0.5 to 0.75. The output variable was the mean of the allele vector of the best student in the population (each individual is a vector of numbers), approximately the average "goodness" of the top student. Figure 8 shows the frequency of the output as a result of these samples; most of the results were within the 0-7 range, with a few outliers. The COMPARE-ON-ONE game chooses the higher of two individuals, thus higher output is better.

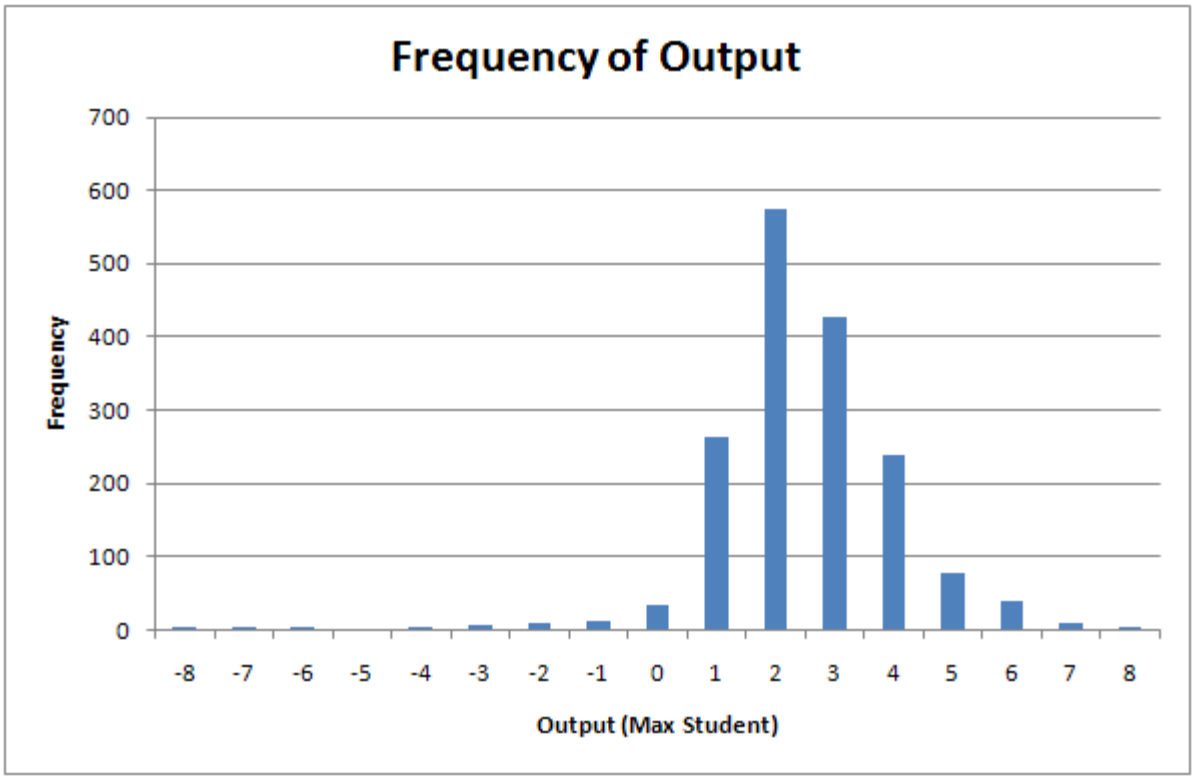


**Figure 8: Histogram frequency of output results of 1700 Latin Hypercube samples.**

Two strong relationships emerged from the analysis. The first is that when both student and test mutation were high, the result was high. The second is that the optimal student archive size depends on the dimensionality of the problem. For problems of low dimensionality, a small archive is best, larger archives produce worse results, for problems of high dimensionality, a large archive is best, larger archives produce better results. Figure 9 and Figure 10 show plots of the samples indicating these results.

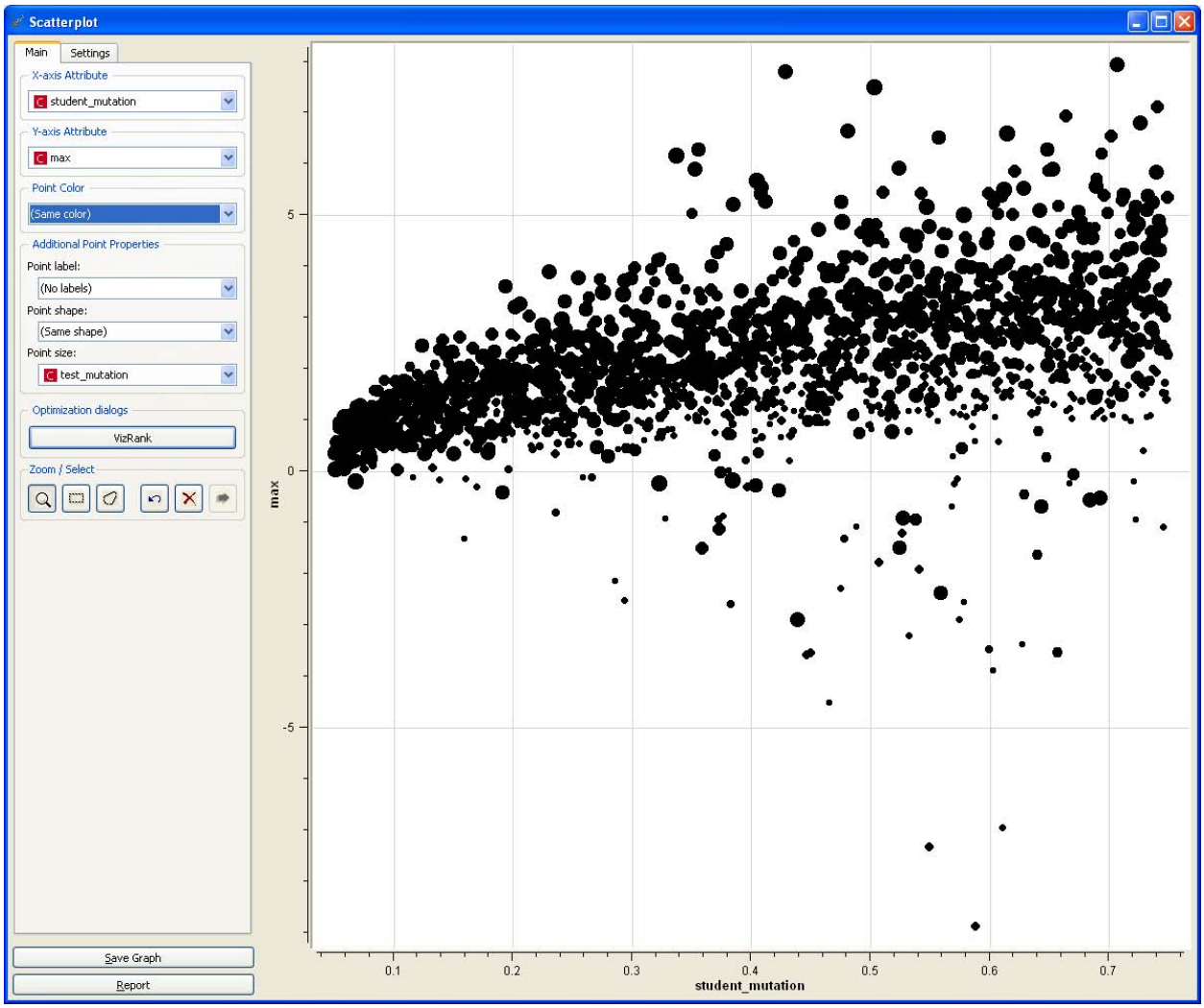


**Figure 9: Student mutation vs. Output. Test mutation determines the size of the circles. Large student mutation (to the right) and large test mutation (larger circles) produce higher output.**

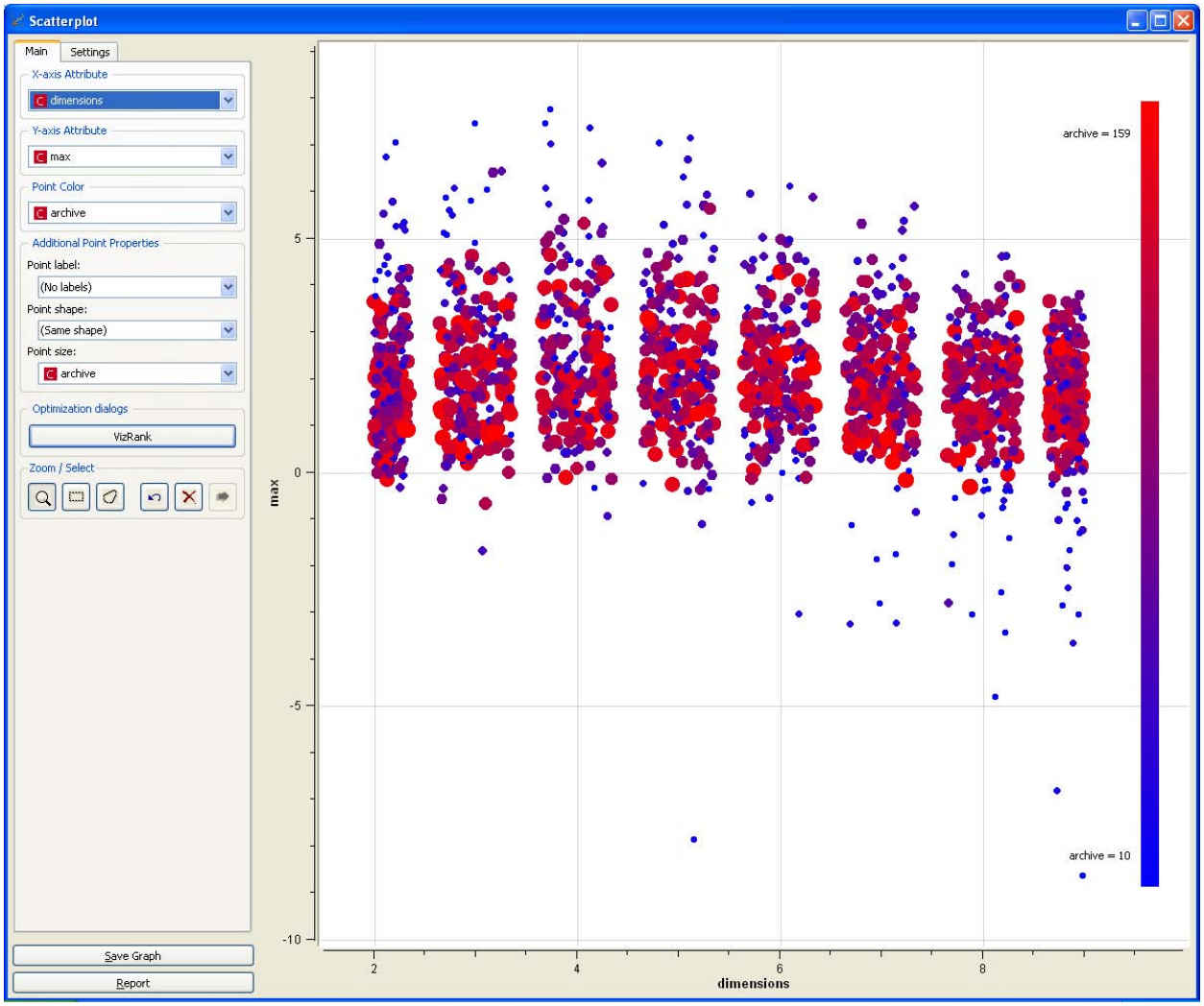


**Figure 10: Dimensionality vs. Output. Student archive size determines the size and color of the circles. Small archives (small, blue circles) are better when dimensionality is small (to the left). Large archives (large, red circles) are better when dimensionality is large (to the right).**

These results translate to two recommendations for selecting parameter values for MaxSolve student-test coevolution. First, the student and test mutation rates should be complimentary. The COMPARE-ON-ONE numbers game benefits in general from a high mutation rate, as mutations do not become more destructive as the individuals improve. However, the improvement in students is limited by both their mutation rate and the ability for tests to detect improvements between mutations. Second, the choice of optimal archive size depends on the problem dimensionality. This is a critical component; larger archive sizes cause more individuals and more computation, increasing running time and resource usage.

With this information, our challenge adaptation and student strategy estimation can be more effective by 1) placing equal emphasis on student and test mutation and 2) estimating the problem dimensionality and tuning the archive size. For example, given a model of the trainee's strategy, ESTATE will generate a new challenge that will defeat the trainee but still fall within

the Zone of Proximal Development (ZPD), allowing the trainee to improve with practice. Using the problem tuned parameters, ESTATE invokes coevolution to improve upon the trainee's strategy until it reaches the edge of the ZPD, and then selects from the latest population of tests to present a new challenge to the trainee. When ESTATE's coevolution is more efficient due to our tuned parameter selection, ESTATE can perform this function for a wider class of skill sets and strategies as well as return results faster and more reliably. Section 1.2 discusses our efforts in modeling trainee strategies.

## 1.2 Task 4: Develop Trainee Model Processing, MoneyBee data analysis.

To assess skills and strategies that people employ when faced with a difficult challenge, we continued our analysis of the MoneyBee data set. During this reporting period we furthered our examination of problem difficulty and began visualization and analysis of player strategy.

### *1.2.1 Problem Difficulty Heuristics*

In the direction of our work on Item Response Theory, we attempted to establish an independent heuristic that could predict the difficulty of a particular MoneyBee problem. Such a heuristic may be able to inform the creation of a Zone of Proximal Development for similar challenge sets to identify challenges which are more difficult but still within the trainee's ability.

Our initial heuristic performs the following calculation to estimate difficulty. Beginning with the initial amount of cents:

1. Remove the odd pennies (modulo five)
2. Search for the solution adding a single coin in a breadth first search (first quarters, then, dimes, then nickels, then pennies), until the problem has only one coin type remaining.

This heuristic makes the assumption that players will attempt larger valued coins first, and that players mentally search for a solution by considering all alternatives in sequence. Because breadth first search is exponential in the number of nodes explored, we take the logarithm of the heuristic as the estimate.

Figure 11 shows the results of plotting the log scaled heuristic against the time taken to complete the problem. As the regression line shows, there is a positive correlation between the heuristic and the time to completion. However, the MoneyBee interface requires players to click on each coin to add it to the total. We hypothesize that this click requires time, problems with lots of coins in the solution will take longer even if the player understands the solution quickly. Figure 12 shows the heuristic value plotted against the time scaled by the number of coins. In this figure, there is a negative relationship between the time per coin and the heuristic. We hypothesize that this relationship is due to the fact that repeatedly clicking on a single coin (e.g. to add 5 quarters) is a rapid action, so that more coins in a problem solution translates to less time per coin.

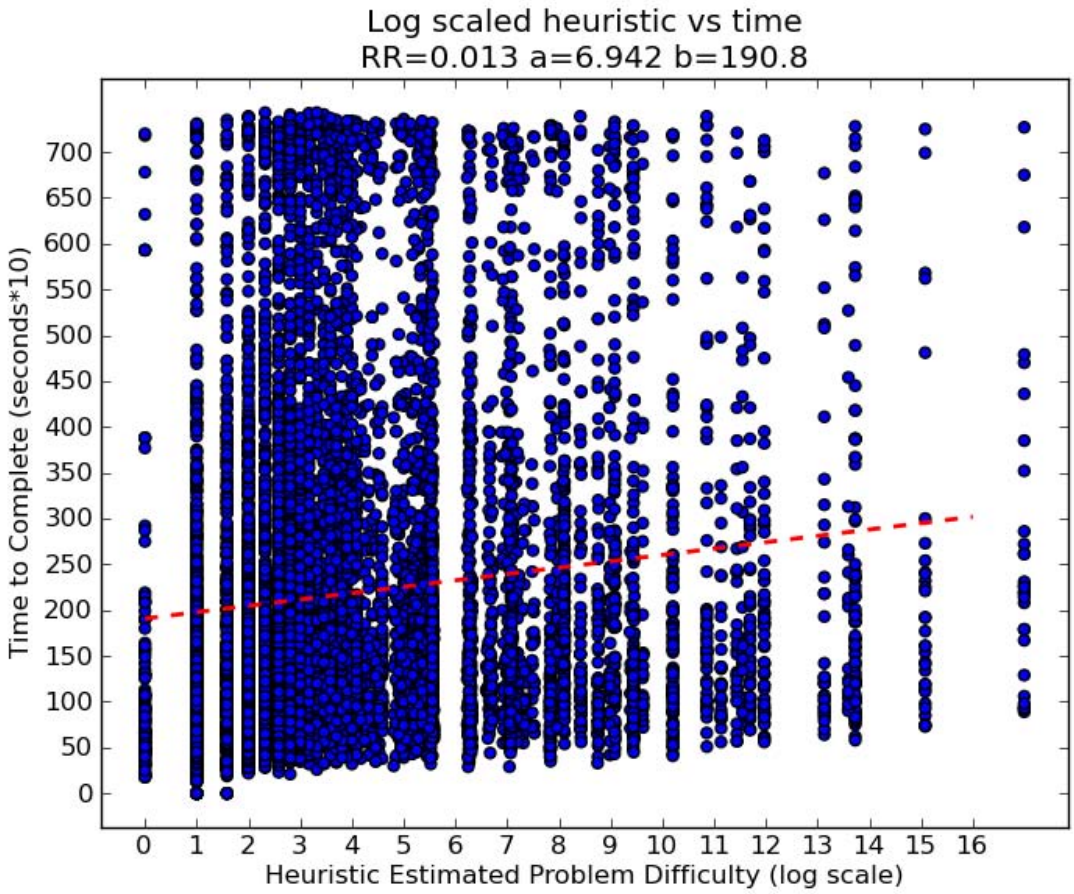


**Figure 11: Log Scaled Heuristic vs. Time to Completion.**

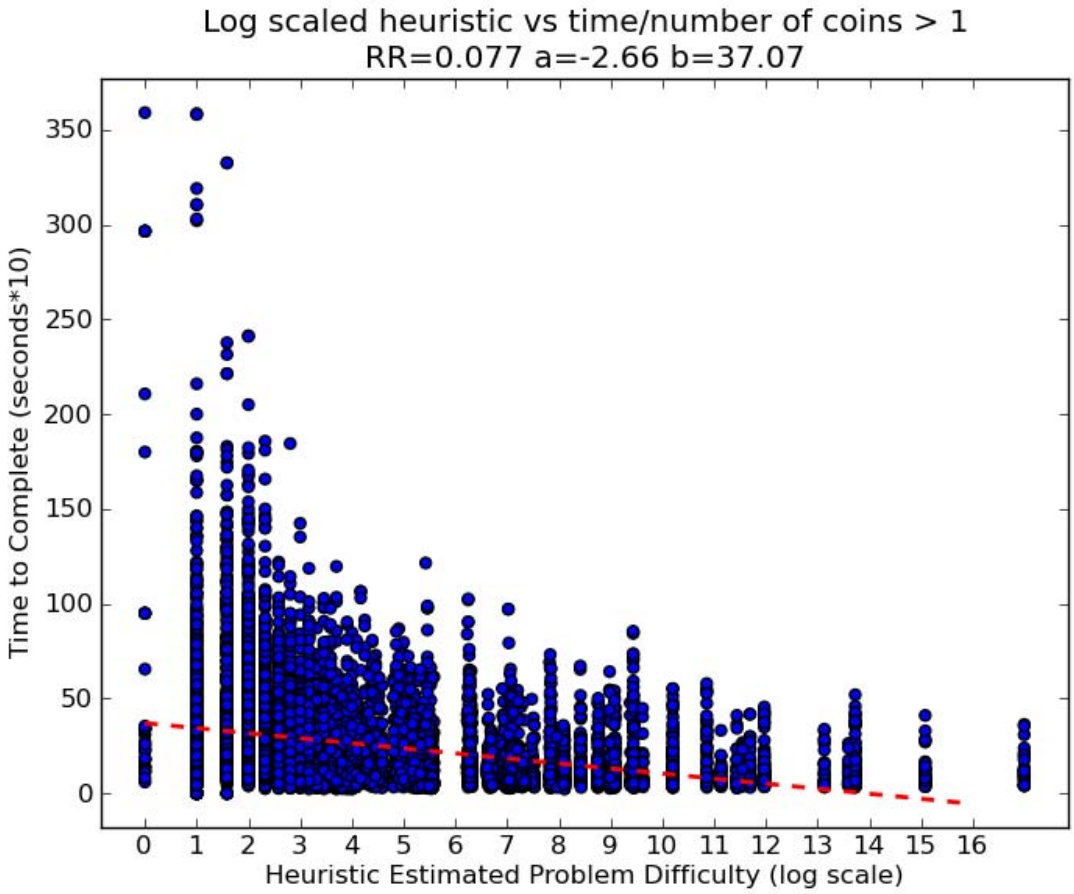


**Figure 12: Log Scaled Heuristic vs. Time to Completion. Scaled by number of coins.**

Our subsequent hypothesis was that switching between coin types (e.g. adding quarters then adding dimes) was causing the players to increase their time. Figure 13 shows the heuristic value plotted against the time per category. Here there is no relationship between the heuristic and the time per category. This result indicates that either 1) the act of entering the solution completely dominates the time the user takes to figure out the solution or 2) the heuristic is not a good estimate of problem difficulty as it relates to time to complete.

We will continue analyzing the MoneyBee data to test these new hypotheses. In particular, we will examine the time per click data more closely to attempt to determine the average time needed to enter the solution as separate from the average time needed to think about the solution. We will also attempt to estimate a heuristic from the data itself, instead of specifying the heuristic a priori according to our assumptions.

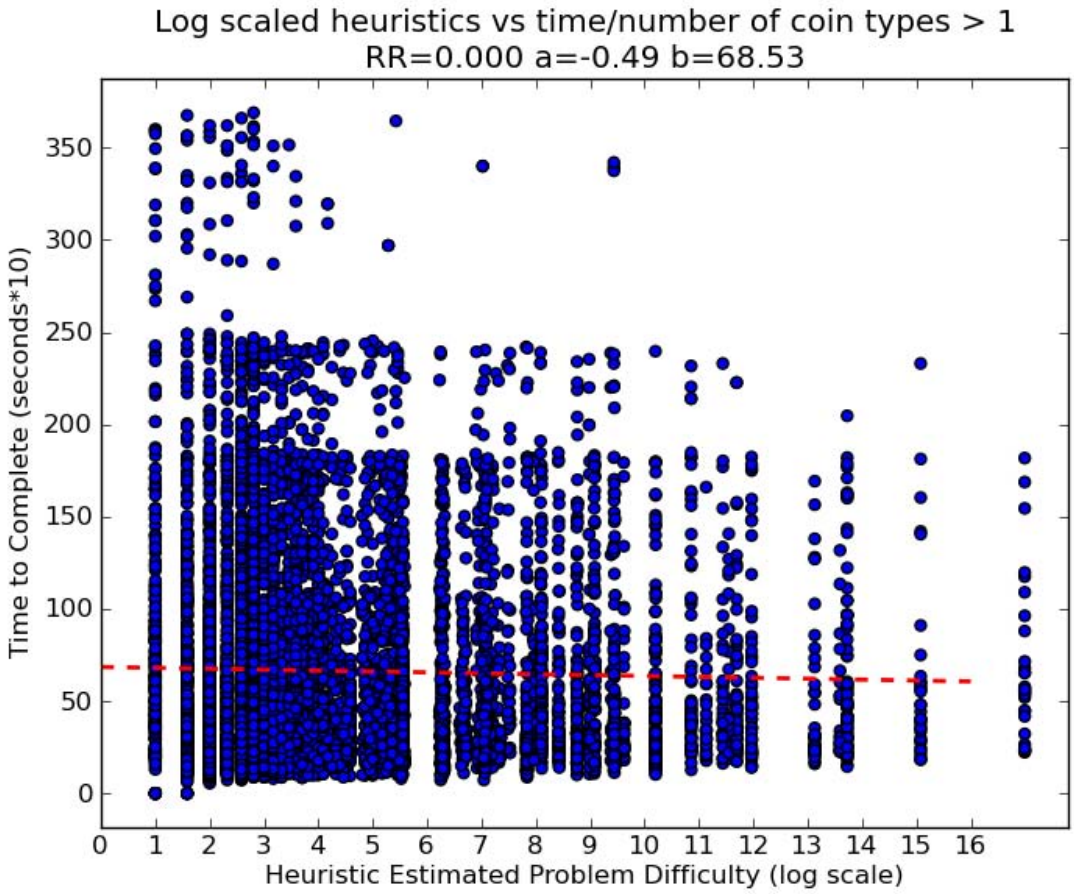


**Figure 13: Log Scaled Heuristic vs. Time to Completion. Scaled by number of coin types.**

### *1.2.2 Player Strategy Visualization*

To improve our understanding of how trainees may employ strategies to approach difficult challenges, we created visualizations of the choices made by players of the MoneyBee game. Our visualizations are graphs of nodes that show how players move through the states of the game by making a choice at each state.

Figure 14 is a close-up view of one such player strategy graph. Each node has 6 fields. The top field is the coin state in the order of quarters, dimes, nickels, and pennies and the bottom fields are the percentages of quarters, dimes, nickels, pennies percentages selected at that state. The top node in Figure 14 is a game state with 1 quarter (represented by 1000), and was arrived at by the selection of a quarter 47% of the time, a dime 41%, a nickel 6%, and a penny 6%. The edges indicate the previous states. In Figure 14, 1000 was followed by 1100 and 1010.

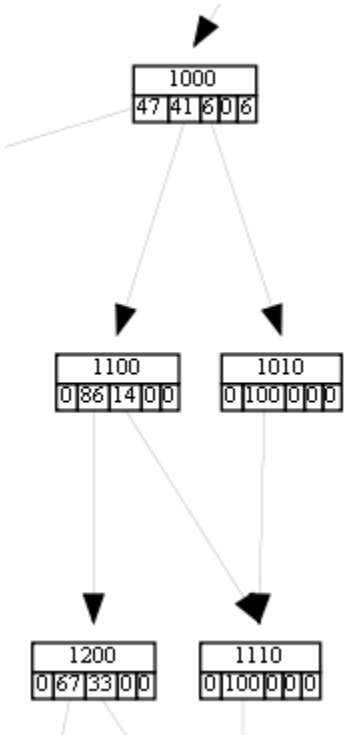


**Figure 14: Detailed of player strategy graph from problem 1332**

Figure 15 shows the player strategy graph of the entire problem of 1322. The two largely disconnected sub graphs indicates that two major strategies have been used on this problem, but one of them is unsuccessful, requiring the player to either backtrack or fail. The node at which these strategies diverge is a key decision point for this problem, and may represent an important concept to practice during training. During the next reporting period, we will work with our academic partners at Brandeis University to improve this visualization (e.g. adding width to the links with higher percentages) and examine the problem set in more detail. This visualization will aid us in targeting quantitative analysis, identifying strategic trends, and presenting the results of our analysis.

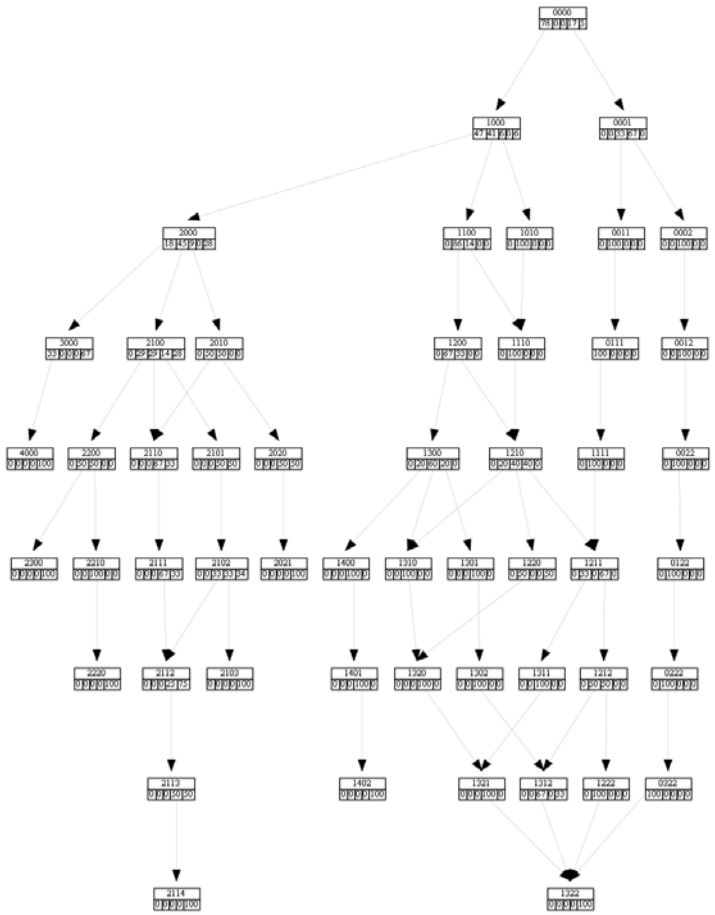

**Figure 15: Entire player strategy graph of problem 1322**

## 2. Reference List


Bouton, C. (1901). Nim, A Game with a Complete Mathematical Theory. *The Annals of Mathematics, 3*, 35-39.

Bucci, A. (2010).
Ref Type: Personal Communication

De Jong, E. (2005). The MaxSolve Algorithm for Coevolution. In Washington, DC.

De Jong, E. & Bucci, A. (2006). DECA: Dimension Extracting Coevolutionary Algorithm. In.

De Jong, K. A. (2004). The Incremental Pareto-Coevolution Archive. In *The Genetic and Evolutionary Computation Conference* (pp. 525-536).

Ficici, S. G. (2004). *Solution concepts in coevolutionary algorithms.* Brandeis University.

Rosin, C. D. (1997). *Coevolutionary Search Among Adversaries.* University of California, San Diego, CA.

## 3. Scheduled Items

In the next reporting period we plan to address the following items:

- Expand our implementation of student-test coevolution to recognize and respond to game *features* instead of simply memorizing game states, thereby expanding the complexity of the games it can solve
- Extend our sensitivity analysis to the Challenge Tree and Nim games.
- Investigate trainee strategy modeling.
- Simulate the loop of 1) trainees attempting challenges, 2) assess trainee skill and strategy, and 3) challenges evolving.
- Continue MoneyBee data analysis in both skill and strategy form.

Sincerely,

Brad Rosenberg
Principal Investigator